SOUND SOURCE ANNOYANCE: THE DETECTION THRESHOLD AND CORE AFFECT FROM PEOPLE WHO ARE ANNOYED.

Bachelorthesis

Pia Burger, s2126532, pia-burger@planet.nl & Daan Veraart, s1480634 , daan@biggus.nl

Abstract: This paper is about 1) a possible sound source in noise detection threshold shift that annoyed people may have developed specifically for the sound they are annoyed by, and 2) how the audibility of an annoying sound influences core affect. The first part of this paper investigates whether people who are experiencing sound annoyance, have developed a lower detection threshold for the annoying sound in comparison with people who don't consider the sound annoying. We expect that people who are annoyed by a particular sound source, are better in detecting the sound, and are capable of hearing this source at a lower SNR. Mixing annoying sounds with pink noise and determining the SNR in which the subjects do not longer hear the sounds tests this expectation. This detection threshold is compared with the detection threshold of subjects who don't consider this sound annoying. We didn't find a significant difference between both conditions. The second part of this study focuses on the appraisal of these annoying sounds. In this part, the annoying sound is added at the end or the beginning of a normal environmental sound, and examined is how the environmental sound is evaluated with and without the annoying sound (as calm, lively, chaotic, or boring). It is expected that the environment is appraised more chaotically or boring when the annoying sound is present. We found a significant difference on the horizontal axis (unpleasant vs. pleasant) on both conditions (adding the sound in the beginning and adding the sound at the end).

Keywords: Detection threshold; Core Affect; Soundscapes; Sound Annoyance; Soundscape

1. Introduction

Soundscapes are important because they affect the wellbeing of people and the quality of life (CALM, 2004). Soundscapes can be evaluated in positive and negative terms, yet they are still poorly understood (van den Bosch & Andringa, 2014). Living in a suboptimal sonic environment can be harmful to the physical and mental health of people, causing stress, sleep disorders and reduced cognitive capabilities (UK Department of Health, 2009 amd WHO, 2011). To improve our understanding of the negative impact of noise on the wellbeing of people, we need to know how people evaluate a soundscape when the annoying sound is audible, and when it isn't. The aim of this paper is to address two topics. We study the detection threshold for one annoying sound of the participants. Earlier investigation showed, that people who are annoyed by a particular sound, state that they are better in detecting that sound than others (Andringa et al, in preparation). Secondly, we analyze the relationship between soundscape appraisal, in terms of pleasantness and eventfulness, and a sound source that's indicated by a participant as annoying.

1.1. Core affect

Earlier research showed that people evaluate soundscapes in terms of pleasantness and eventfulness (Axelsson et al., 2010). Russell (2003) states that people use dimensions like pleasantness and activation to describe their moods. These dimensions can be integrated in a single concept 'core affect'. Kuppens et al. (2012) showed that the appraisal and core affect mutually influence each other. So we might propose that this is also the case for soundscapes and moods, and that both affect each other. Andringa and Lanser (2013) proposed four qualitatively different types of soundscapes: lively, calm, chaotic and boring. Figure 1 shows these four soundscapes types, which can be

classified according to their eventfulness and pleasantness.

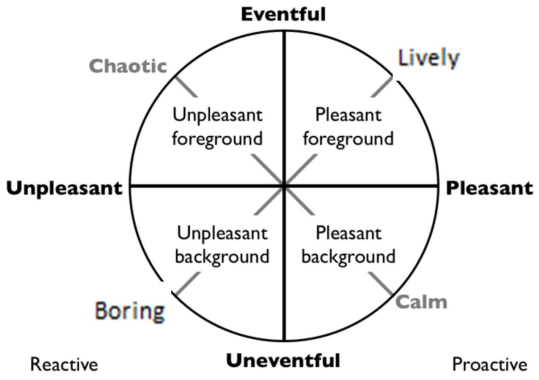


Figure 1: Four types of soundscapes (chaotic, lively, boring, and calm). Basic dimensions are eventfulness vs. pleasantness. (From Andringa&Lanser (2013))

1.2. Expectations

We have a number of expectations that can be tested with the data that we have acquired:

- People who are sound annoyed, have a lower detection threshold for the annoying sound than people who don't consider this sound annoying.
- The addition of any annoying sound to a natural soundscape, contributes to a chaotic (or possibly boring) interpretation of the sonic environment.
- Sounds, such as bird song, that connect us with a natural environment contribute to a calm interpretation of the sonic environment.

2. Methods

2.1. Participant

We approached 150 participants of a previous research (Andringa et al, in preparation) via an e-mail with the results from this previous research in which we asked them to participate again. For our research we needed people who are highly annoyed by one or more sounds. That's why we referred in this e-mail to the site www.auditoryenviroments.org, where all the results of this previous research were explained and where the results were presented in such a way as to motivate possible participants. That's why you can find three movies on this website, made by us, where everything is explained in simple language. 12 participants responded that they wanted to participate in our research. Another 10 participants responded to our mail, but were not able to participate because of the lack of suitable recording equipment. Of the 12 participants, were four woman, and eight men. The participants were between 30 and 60 years old.

Of the 12 participants, all persons were able to perform experiment 2. They understood what to do, and responded in the expected way. Three of the participants didn't really understand experiment 1, because they thought the pink noise was traffic noise, and so they always responded that they heard 'something'. The participants who responded like this, were really annoyed by traffic noise.

2.2. Demographics

Because we conducted the experiment at the home of that participants we selected only participants who live in northern Netherlands (Friesland, Groningen, Overijssel, Flevoland, Gelderland, Drenthe) to minimize travel time. The average travel time per participants was 3 ours, and the two experiments lasted about one hour (inclusive listening to the stories they told)

2.3. Procedure experiment 1

In the first experiment aim to determine whether annoyed people have a lower detection threshold for the annoyed sound compared to sounds that aren't annoying. We presented 3 target sounds (of which only one is their annoying sound), embedded in pink noise. We presented a sequence of targets in noise in which we varied the Signal to Noise Ratio (SNR). In each presentation the participants had to indicate whether they could detect the target. If they could, the added sound's volume was lowered by for example 3 dB (SNR reduced by 3 dB). If they could not detect the sound the SNR was raised. If they responded that they didn't detect the sound this time, the trial was over. If they did hear the sound, the next dB level of the sound, is between the dB level they didn't hear, and the dB level they did hear. Figure 2 visualizes this process.

To prevent spurious detections, there is a 20% probability that with the next step, the sound

isn't added to the pink noise, after a participant responds for the first time that he doesn't hear the sound. Before the experiment started, participants are told about the possibility of the sound not being present in the pink noise.

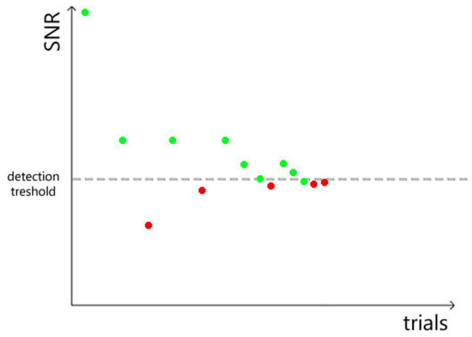


Figure 2 This figure shows the process for experiment 1. The green dots indicate that a participant did hear the sound, whereas the green dots indicate that the participants didn't.

2.3.1 Equipment

This experiment was run on a laptop and presented to the participants using a customized Matlab graphical user interface (figure 3.) The stimuli were presented with a closed headphone (Sony HD 150) (which is connected with a cable to the laptop) at comfortable listening level well above the ambient noise level.

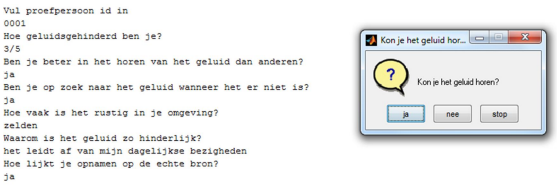


Figure 3 Matlab interface

2.4. Procedure experiment 2

In the second experiment we want to determine how the addition of an annoying sound to a normal environmental sound influences the appraisal of the environment. We developed an Android-app for this experiment, which displayed a simplified version of Figure 1, see appendix 1. The participants had to indicate by indicating on the tablet how they evaluated the environment in terms of eventfulness and pleasantness. We gave participants a short description of what they saw, and started with a testing trial, to see if they understood what they had to do. This test-trial consisted of a one-minute audio clip of a natural environment (a forest with birdsongs), combined with 3 of the following sounds:

1. jogger, children playing, humming woman,
2. chainsaw, lawn mower, train
3. owl, turkey, woodpecker

The test-trial consisted of a fixed sequence and fixed combination of the sounds above. The sounds where introduced at the 5, 25 and 45 second mark. After the test trial the participants were presented with a second soundscape where after 30 seconds the sounds the participant classified as annoying, is introduced. The annoying sound lasted till the end. In the third 1-minute audio clip the, annoying sound is added at the beginning and lasts until the 30 second mark. You can find a visualization of this process in figure 4. The order of the three clips is fixed, and we chose this sequence, because the second clip aims to investigate how much time it takes to influence the participants appraisal. The third clip is used to study how long it takes to change the appraisal of the sonic environment after the annoying sound stopped. We assumed that people could have become irritated from the second clip, and that this may influence the third clip. If we were to reverse the order, we would give a distorted picture.

Figuur 4 Visualization of the trials presented to the participants. Number 1 shows the sound immediately presented after the start of the trial. Number two represents the trial where the sound is added after 30 seconds, which will last till the end.

The Android-app records the movement of the finger of the participant over the screen on the diagram. The axes of the diagram are normalized from -1 to 1. Movement is recorded by time in seconds and milliseconds.
All participants do three trials with the sound that annoys them and two times three trials with sounds that annoy other participants.

2.4.1 Equipment

This experiment was run on an Android tablet and presented with a self-written android app. The app can be downloaded at: http://interneteindbaas.com/coreaffect Again, the stimuli were presented with a closed headphone at comfortable listening level well above the ambient noise level.

3. Results

3.1. Experiment 1

To determine whether there's a difference between the detection threshold SNR for the sound people are particularly annoyed by, and the detection threshold SNR for the sounds they aren't particularly annoyed by, we compared the means of these two groups. Figure 5 shows the differences between the two groups. This figure shows a difference in mean between the two different groups. There is not a significant difference between the two different groups ($F(1,14)=-0.732$, $p>0.05$).

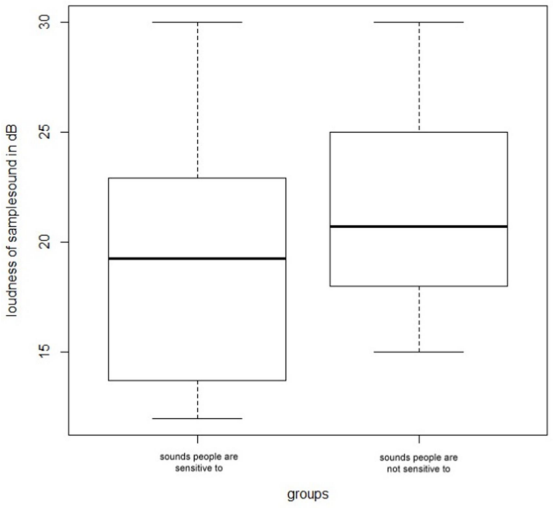


Figure 5 This figure shows the difference between the mean of the SNR for the sound people are particularly annoyed by (left boxplot), and the mean of the SNR for the sounds they aren't annoyed by (right boxplot).

3.2. Experiment 2

For experiment 2, we focus on a couple of things:

- The size of the movement when the annoying sound is added after 30 seconds (called condition 2.1), or removed after 30 seconds (called condition 2.2).
- The final coordinate for the x-as (unpleasant vs. pleasant), 5 seconds after the annoying sound is added (condition 2.1) or removed (condition 2.2).
- The final coordinate for the y-as (eventful vs. uneventful), 5 seconds after the annoying sound is added (condition 2.1) or removed (condition 2.2)

3.2.1 Condition 2.1

Condition 2.1 consists of the (annoying) sound that is added to a normal environment (forest sounds), 30 seconds into the 60 second recording. A typical response to this condition looks like the one in figure 6. The response to participant's annoying sound is colored in red. After the sound is added, participants respond strongly to their own annoying sound and not to sounds that annoy others (green). Appendix 2 provides an overview of all the responses.

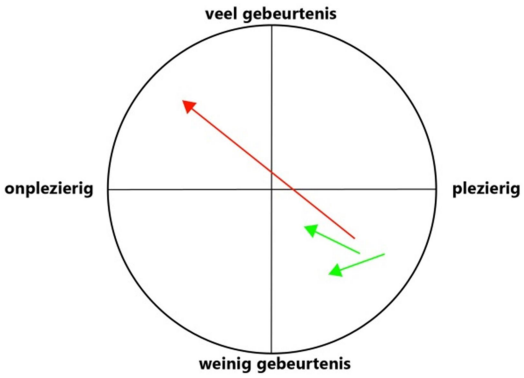


Figure 6 This figure shows a typical response of the participants for condition 2.1, whereby their own annoying sound is colored in red, whereas the other sounds are colored in green.

The mean distance for the annoying sounds versus the not-annoying sounds, are drawn in figure 7. There is a significant difference between the two groups for this condition ($F(1,16.1)= 3.1$, $p<0.05$).

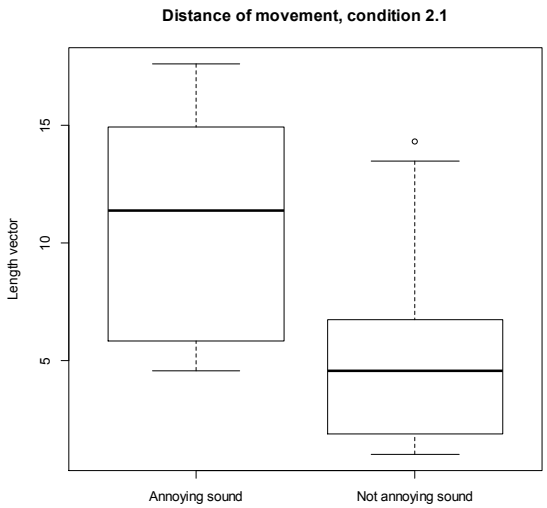


Figure 7 This figure shows the difference between the mean distance of the sound people are particularly annoyed by (left boxplot) and the mean distance of the sounds they aren't annoyed by (right boxplot).

Another way to judge the differences of the (not) annoying sounds, is to look at the final coordinates, where the participants end up 5 seconds after the (annoying) sound is presented. We've looked at the differences in the X-as (unpleasantness vs. pleasantness) and the Y-as (eventfulness vs. uneventfulness). The X-as is scaled from -10 to 10, and the Y-as is scaled from 10 to -10. There is a significant difference between the sounds on the X-as ($F(1,27)=-4.3363$, $p<0.05$), whereas there isn't on the Y-as ($F(1,17)=0.13$, $p>0.05$).

3.2.2 Condition 2.2

Condition 2.2 consists of the (annoying) sound being removed after 25 seconds from the normal environment. Figure 8 shows a typical response. The arrows show the difference in evaluation of the environment before and after removing the sound, with an interval of 5 seconds. There is an overview of all the responses in Appendix 3.

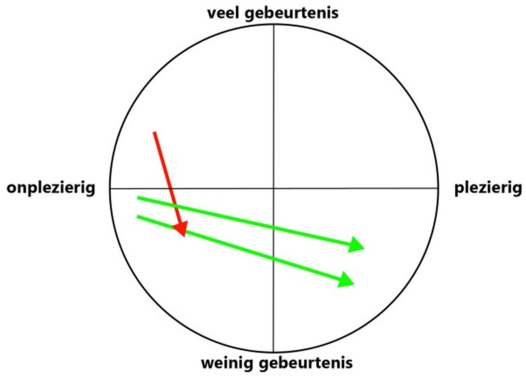
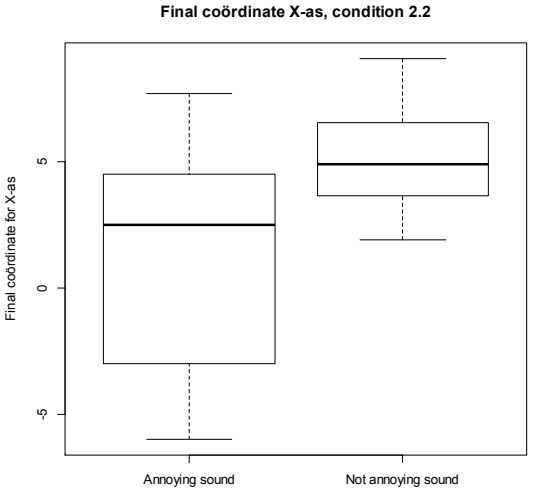


Figure 8 This figure shows a typical response of the participants for condition 2.2, whereby their own annoying sound is colored in red, whereas the other sounds are colored in green.

Again, we looked at the difference of the mean distance between the annoying sounds versus the not annoying sounds. There is no significant difference between the two groups in this condition ($F(1.19,6)=1.15$, $p>0.05$).

Just as with condition 2.1 there is a significant difference on the X-as ($F(1.19,6)=1.15$, $p<0.05$), but not on the Y-as ($F(1.19,6)=1.15$, $p>0.05$). The difference on the X-as is drawn in figure 9 (see below).



.

Figure 9 This figure shows the difference between final coordinate on the X-as of the sound people are particularly annoyed by (left boxplot) and final coordinate of the sounds they aren't annoyed by (right boxplot).

3.2.3 Overall comments of the participants

It has to be noted that in cases where there are two strong reactions, the participants noted that

the other sound was also one that they were hindered by in the past or they experienced that the sounds where similar. People where the reactions are similar remarked that they disturbance of the forest sound was the thing that annoyed them particularly. One participant interpreted the wind in the trees as cars, which resulted in a situation where the entire recording was unpleasant. Another person indicated that hearing voices of children annoyed him, because one of his own children was bullied when she was young, so hearing the voices triggered that event happened in the past. It was such a strong reaction to that sound, that he was annoyed for the rest of the fragment.

4. Discussion

4.1. Experiment 1

Experiment 1 didn't show a significant difference between the SNR for the sound people are particularly annoyed by, and the SNR for the sounds they aren't annoyed by. There's an tendency that the SNR for the first group is lower, but it may be because of a lack of subjects, that there isn't a significant difference. More than half of the subjects are better able to detect the sound they are annoyed by in comparison to the sound they don't think is annoying (i.e. the SNR decreases more for their own annoyed sound in comparison with the other sounds).

There are a few reasons why this experiment doesn't have the expected results; 1) Some subjects (5 participants) had to be excluded of the experiments in advance, because they are so triggered by any sound, they would have been physically ill if they would had participated (the pink noise itself would had been the reason itself to become ill), so they contribute to the tendency, because we had to exclude the most sound annoyed people, and 2) Because many annoyed people (there were 6 persons who were able to participate, but who couldn't because the noise wasn't hearable when adding to the first or second experiment) suffer from low-frequency sounds (such as windmills or air conditioning), that the adding these kind of sounds to the pink noise, made them inaudible.

Subjects who participated and were annoyed by traffic sounds, thought the pink noise itself was traffic, so they always indicated to hear the sound (even if there wasn't a sound mixed with the pink noise), which lead to data which isn't correct anymore. A second trial for this experiment would be useful, if there are more subjects, who aren't annoyed by traffic sound or other low-frequency sounds.

4.2. Experiment 2

In the first condition of experiment 2, we found two significant differences: the distance of the movement people made when there was an (annoying) sound added, and the final coordinate on the X-as after 5 seconds the added sound started. As can be concluded from Attachment 2, some of the participants reacted more extreme when they heard their sound then when they heard other people's sounds. In other cases the movement is similar.

The significant difference between the two different sound groups (annoying sound versus not annoying sounds), indicate that people evaluate their environment more unpleasant, and react more heavily when they hear their own annoying sound in comparison with sounds they aren't annoyed by. Also, there seems to be something we can call "the fairness effect" (Maris et al., 2007). People seems to be very subjective according to the evaluation of their environment; when two sounds are acoustically very similar, but for one of the sound sources they think it's fair to be there, then they evaluate the environment different, even if the sound is the same, or the one they think is fair to be there is louder. An example is that people who live between TT-Assen and the shooting range, where the sounds derived from the shooting range are louder and more intense, aren't annoyed by the shooting range, but are from TT-Assen. They indicate that TT-Assen is a hobby, whereas the shooting range isn't (it's for military who have to practice). So they seem to give an evaluation to the sound, regarding to the association they make with the sound they hear (is it fair and explicable for the sound to be there and to be as loud as it is).

In the second condition of this experiment, we found 1 significant difference: the final coordinate on the X-as, 5 seconds after the (annoying) sound was removed. People seems to

be hindered, even after the annoying sound is removed, so the sound still influences their appraisal, even when it isn't audible. In Attachment 3, you'll see that although the sound that annoyed them was gone, some people still qualified the sonic environment as unpleasant. Whereas when they were presented with another sound that disturbed the forest sound they qualified the sound as pleasant within five seconds after that sound stopped. This indicates that the sound they are annoyed by creates an environment that even when the sound is inaudible they are still affected by. For example figure 6 shows that when the annoying sound of the participant stopped, he qualified the sound as less eventful, but it didn't cross the pleasant 0 point, whereas when the same happened with the other sounds, he didn't focus on reduced eventfulness but went almost completely into the direction of pleasant.

Participants indicated that they were afraid the annoying sound showed up again, whereby they didn't evaluated the environment as pleasant.

5. Conclusion

As previously mentioned, in earlier research participants noted that they thought they are better in detecting their annoying sounds then others. We can't say that's true, because there wasn't a significant difference. But what we can say, is that adding or removing an annoying sound to a normal environment, gives different responses than adding or removing a sound that isn't annoying for a particular person. Adding an annoying sound makes people to evaluate their environment as unpleasant, whereas people who don't think the sound is annoying, don't respond at all, leaving the evaluation on the pleasant side of the core affect. Removing an annoying sound, doesn't make a person to evaluate their environment as pleasant right away, but seems still to influence the evaluation of a normal environment as unpleasant. People seem to be unable to forget their annoying sound, if they just heard it.

6. Who did what?

You can find the timeline and the division of the different tasks to perform in Appendix 4.

References

Axelsson, Ö., Nilsson, M.E., & Berglund, B. (2010). A principal components model of soundscape perception. *The Journal of the Acoustical Society of America,* 128(5), 2836-2846.

CALM. (2004). *Research for a quieter Europe in 2020.* European Commission Research Directorage-General.

Kuppens, P., Champagne D., and Tuerlinckx F., "The Dynamic Interplay Between Appraisal and Core Affect in Daily Life," *Frontiers in Psychology* 3 (2012): 1–8, doi:10.3389/fpsyg.2012.00380.

Maris E., Stallen P., and Vermunt R., "Noise Within the Social Context: Annoyance Reduction Through Fair Procedures," *Journal of the Acoustical Society of America* 121, no. 4 (2007): 2000–2010.

Russell, J.A. (2003). Core affect and the psychological construction of emotion. *Psychological Review.*

UK Department of Health (2009). Environmental Noise and Health in the UK. *Report by the Department of Health Ad Hoc Expert Group on Noice and Health, Dept. of Health, London.*

Van den Bosch, K.A., Andringa T.C., (2014). The effect of sound sources on soundscape appraisal. *11th International Congress on Noise as a Public Health Problem, Japan.*

Van den Bosch, K.A., Burger, P., Andringa, T.C., (in preparation).

WHO, *Burden of Disease From Environmental Noise*, (Bonn: World Health Organization, 2011).

Appendix 1

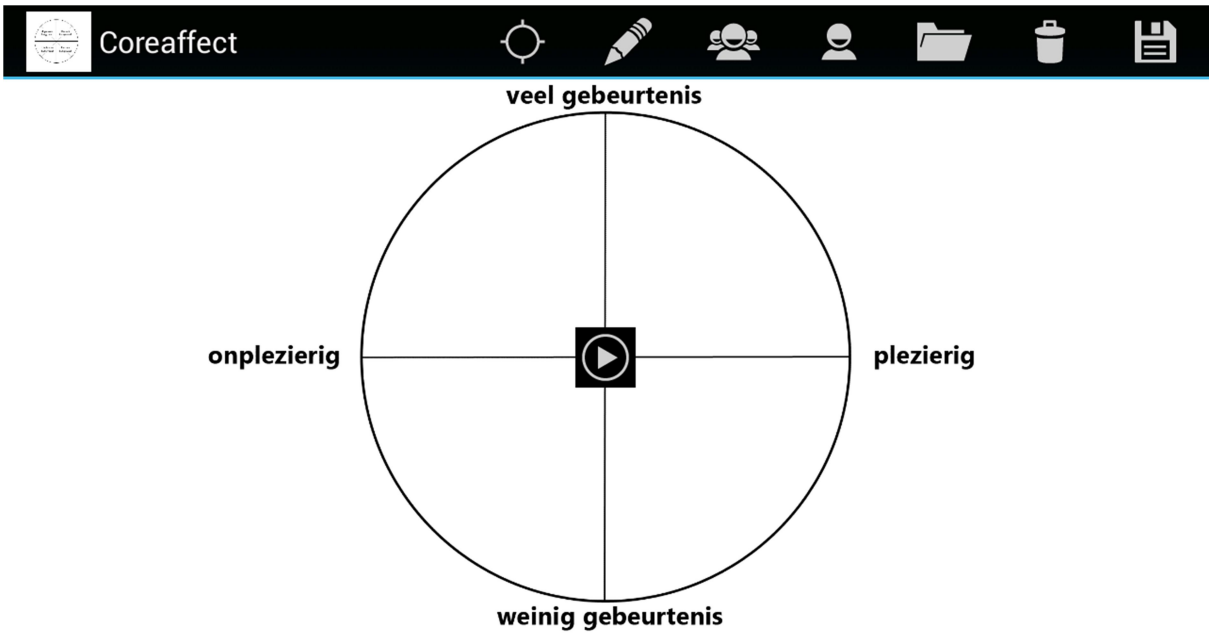


Figure 10 Screenshot of the Android application

Background:
For the second experiment the goal was to design a way to capture the core affect of certain sounds. The method used before consisted of a joystick that could be used to navigate an indicator around in the circle. The disadvantage of that was that the joystick has a build in tendency to go back to the middle, this might influence the result and so an alternative had to be designed.

Design requirements:
- The user has to be able to indicate on the core affect circle how he experiences the heard sound.
- The way to do this must be intuitive.
- The way to do this must not be influencing the results.
- There must be visual feedback that in fact something is happening.
- The information gathered from the user indicating the core affect must be linked to the sound that is heard.
- The information must be saved in an easily accessible format so it can be further analyzed.

Platform choice:
The first three design requirements where the most influential for picking the platform to build the application on. The feedback about the joystick, let to the choice of a touchscreen, as it had the same ease of use, but not the disadvantage of steering the user into a certain direction. Four types of touchscreen where considered: Smartphone, tablet, touchpad and laptop with touchscreen. The smartphone fell of as its size might lead to imprecise results. Of the remaining three the tablet appears to be most intuitive and easy to use. There was chosen for an Android tablet because of the availability and versatility of the operating system. The Android application then was designed after the latest standards set by Google, to make the application look the most natural.

Interface:
The application (see figure 10), made for the second experiment, consists of the Android action bar seen at the top, which holds 7 or 8 buttons (depending on the state of the application) and the main area where the graph is drawn. Table 1 explains the functions of the buttons. In the main area the graph you see is drawn and centered on the screen. On top of that there is an invisible layer that detects the touch input from the user. This is done in two layers to make sure there is no strange cropping or changes in the aspect ratio or quality. Overlaying on top there is a play button. Pressing the play

button will start the loaded mp3 file and make the play button disappear, so the participant can use the entire graph to indicate his or her experience.

Code:

interface:

The layer, or View, that detects the user input, gets initialized when the application is started. On initializing it will take the height and width of the screen the application is running on and calculates the part of the screen that constitutes the circle of the graph. In this circle a Canvas is made along with a Paint element so the user can draw inside the circle. The paint element was added so the user gets visual feedback that he is moving his or her finger. This was a design requirement and added because if there is no feedback added using this paint element, the user gets an unjust feeling that nothing is happening, which in turn can influence the results. When touching the main part of the screen the application adds the coordinates of the place the screen was touched to a list of coordinates. To make the application consistent over all different screen sizes, the coordinates are recalculated so the center of the graph has coordinate (0, 0), and the axis run from (-10, 0) to (10, 0) and (0, -10) to (0, 10).

drawing:

Based on two settings (draw circle and draw trace) the original coordinates may also be added to the bottom of two stacks. If the draw trace functionality is on, whenever the user touches the screen within the circle a line is drawn following his/her movement. That touch now also starts a timer that triggers a function half a second later. That function takes the coordinate from the top of the stack and erases the paint at that coordinate. This will create a temporary line when the user moves his finger inside the graph. The other setting, draw circle, draws a circle around the point the user touches the screen. Only the part of the circle that falls in the graph is drawn. Before the circle is drawn, the last circle is erased, making sure there is only on circle visible at any time. When the user stops touching the screen, the circle is also removed. The process of erasing the paint either after a set time of when the next circle is drawn is implemented because Android has no native way to temporarily draw something and although the visual feedback was a requirement, if the paint stays on screen the screen gets cluttered which negates the effect of not influencing the results.

mp3 functionality:

To properly align the user input with the sound the user is hearing the application plays the sound itself. The open file dialog allows the user to choose a specific mp3 file. After choosing an mp3 file the play button can be pressed to start it. Pressing the play button without first choosing an mp3 file will give a warning informing there was no sound chosen. When the play button is pressed the sound starts and the exact moment at that time is saved. The list with coordinates includes the time in milliseconds between the exact moment the sound started and the moment of the touch event. Not only will pressing the play button remove the play button from the screen, it will also add a stop button in the top of the screen. Pressing the stop button will stop the sound file prematurely. After the sound has ended, the stop button disappears and the play button reappears.

Data collection:

As easily accessible data format a simple text file is used. Pressing the save button will save all the recorded data to a plain text file onto the main storage of the Android tablet. The text file is named by the following template: <year><month><day><hour><minute><second># <name experimenters> – <id of the participant>_<name of the sound file>.txt (without the brackets). Where the time is the time the file is saved. This to make sure, there are no duplicate filenames and the files are easy to sort and find.

In the file the name of the experimenters, participant id and name of the song are again listed, as well as the starting time of the played file, the end time, and the total time in milliseconds the song played. Beneath that the list of coordinates is printed, in the format: <hour><minute><second> [<time in milliseconds after the sound started>] -> <adjusted X coordinate> - <adjusted Y coordinate>. After the file is saved the list of coordinates is emptied. To make sure there is no data loss by

saving twice in short repetition, the save button is disabled for 3 seconds after every save. Guaranteeing there will be a second file with a different name, so no data is lost.

Table 1 List of buttons and their functions

	Toggle draw circle, turns the option to draw a circle around the finger on/off		Clear list, clears the current list of coordinates
	Toggle draw trace, turns the option to draw a trace after the finger on/off		Save, saves the current list of coordinates
	Enter the name of the researchers, opens a dialog to enter the names		Play, plays the opened mp3 file, and set play time. (only visible while there is no sound playing)
	Enter the id of the participant, opens a dialog to enter the id		Stop, plays the opened mp3 file, and set play time. (only visible while there is a sound playing)
	Open mp3, opens a file dialog to choose a mp3 file from the SD card of the device		

Development:

The development is done in three stages: Drawing, complete interface, MP3 and data collection. Every stage is tested and evaluated before the next stage begins. The drawing stage consists of building the functionality of drawing on the screen. In this stage a debug section was added to the left of the screen to check if the data recorded is accurate and adequate. For an example of this setup see figure 11. After tests confirmed this was the case, the next step complete interface consisted of making the interface to the end product, a new clearer diagram was created, the debug information removed and the buttons added in the action bar so they wouldn't clutter the screen. After the interface was tested and approved the last stage of adding the sound and save functionality was added. This was done at the last step, because of the testing for this step was unreliable in the emulator and extensive live testing is the easiest with a completed interface.

Testing:

After the first two stages, testing is done by testing just the functionality at that point, this creates the risk that at the last stage extra requirements will be found, but it is the most efficient way. During testing two extra requirements where added: The circle showing the location where the screen is touched at that exact moment and an updated clearer graph. Both where implemented in the second stage.

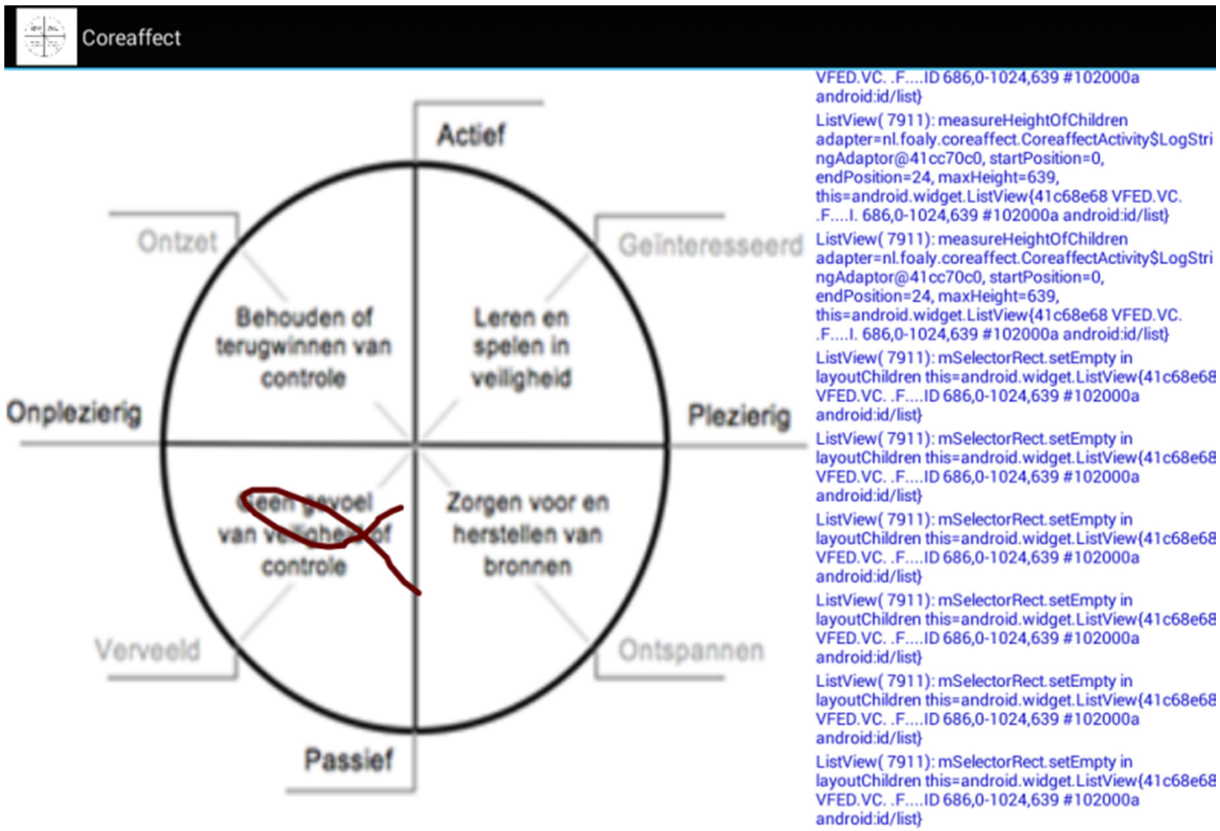
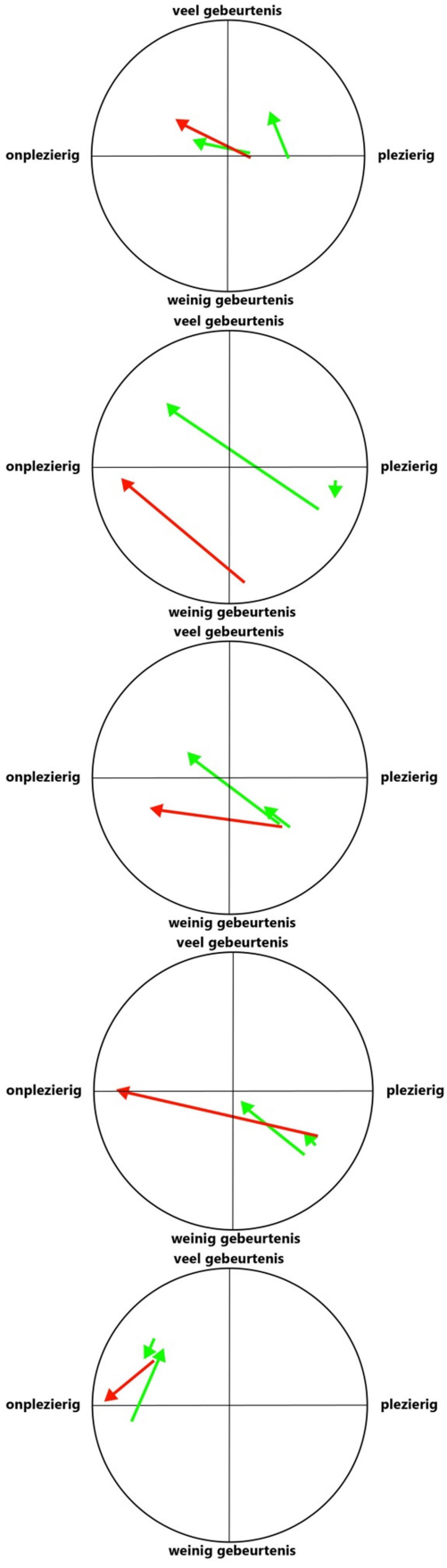
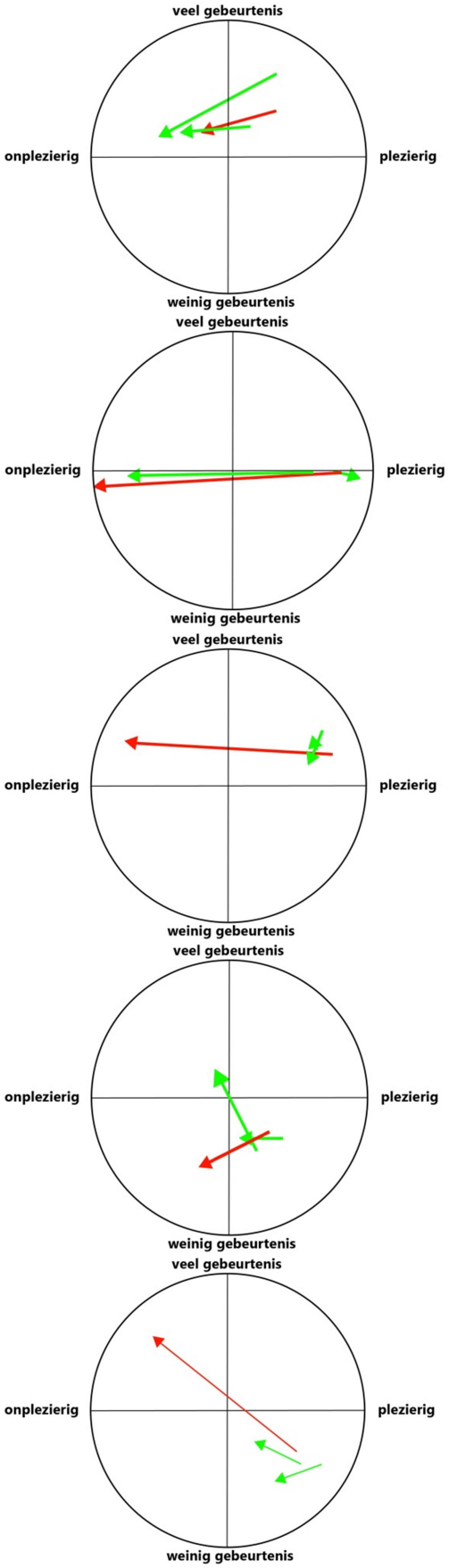
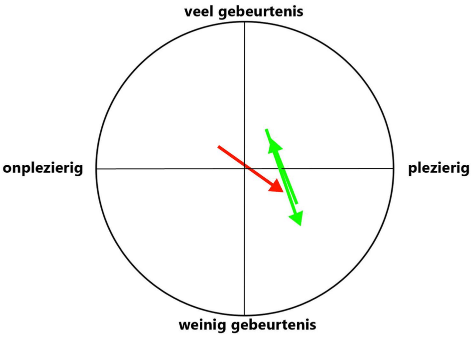


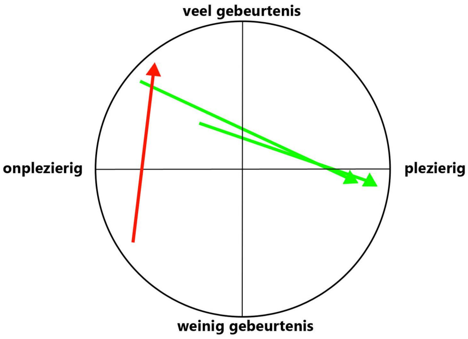
Figure 11 Screenshot of the Android application at stage 1

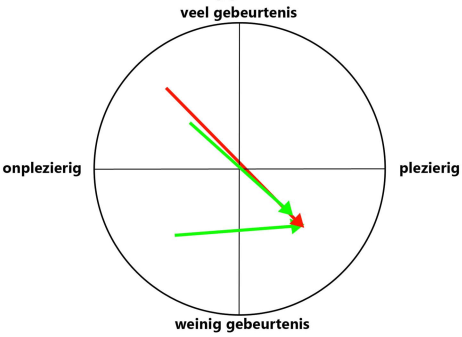
Appendix 2

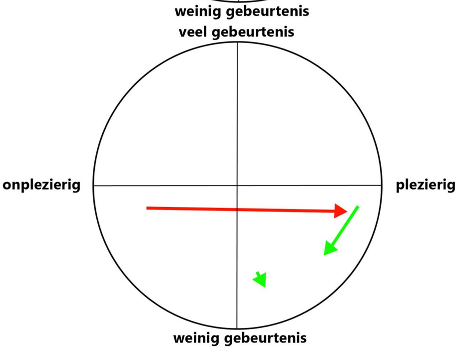

veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis
veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis
veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis
veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis
veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis

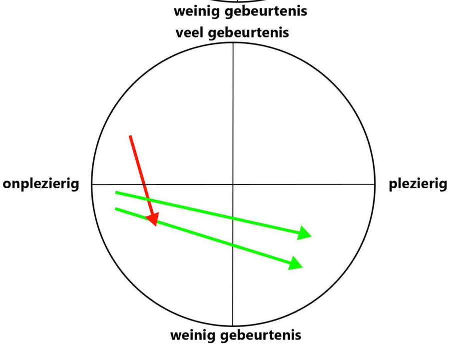

veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis
veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis
veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis
veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis
veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis

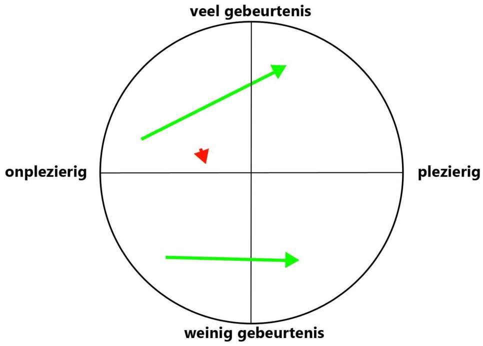

veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis

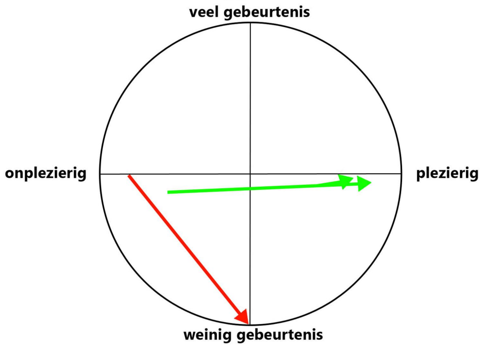

veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis

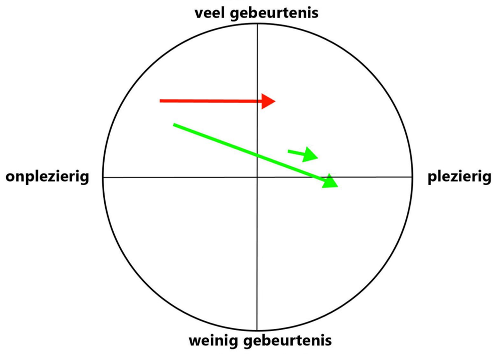

veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis

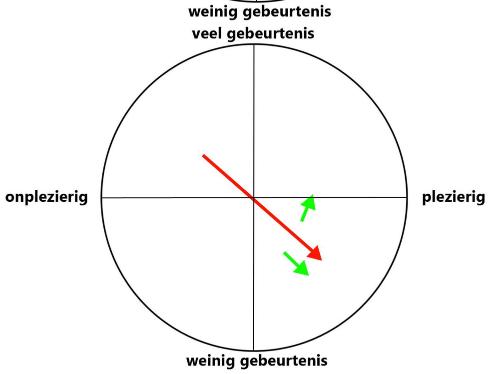

veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis

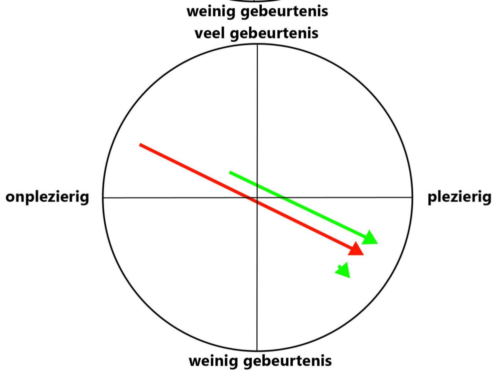

veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis


veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis


veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis


veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis


veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis


veel gebeurtenis
onplezierig
plezierig
weinig gebeurtenis

Appendix 3

The Java for Android source code:

```
---------------------------------------------------- coord.java ----------------------------------------------
package nl.foaly.coreaffect;

//Object to store the coördinates of the screen
public class coord {

	float x;
	float y;

	coord(float f, float g){
		this.x = f;
		this.y= g;
	}
}
```

```
---------------------------------------------------- CoreaffectActivity.java ----------------------------------------------

package nl.foaly.coreaffect;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import android.app.AlertDialog;
import android.app.ListActivity;
import android.content.Context;
import android.content.DialogInterface;
import android.graphics.Color;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.text.format.Time;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;


public class CoreaffectActivity extends ListActivity {

    private LogStringAdaptor adaptor = null;
    private ArrayList<String> logarray = null;
    private LogReaderTask logReaderTask = null;
    private DrawingView drawView;
    private boolean log = false;
    ArrayList<String> trace = new ArrayList<String>();
```

```
final MediaPlayer mp = new MediaPlayer();
String m_chosen = "";
Time now = new Time();
Menu menu;
Handler timeout = new Handler();

//on create, declare the canvas that will be 'painted on'
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   drawView = (DrawingView)findViewById(R.id.drawing);
}

//create menu, to make menubuttons pushable
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.menu, menu);
   this.menu =menu;
   menu.findItem(R.id.stop).setVisible(false);
   return super.onCreateOptionsMenu(menu);
}

//add action to menu buttons
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   // Handle presses on the action bar items
         View view;
   switch (item.getItemId()) {
      case R.id.save_action:
         view = findViewById(R.id.save);
          save(view);
         return true;
      case R.id.draw_action:
         view = findViewById(R.id.drawtrace);
         toggleTrace(view);
         return true;
      case R.id.onderzoeker_action:
         view = findViewById(R.id.onderzoeker_action);
         setOnderzoekers(view);
         return true;
      case R.id.participant_action:
         view = findViewById(R.id.participant);
         setName(view);
         return true;
      case R.id.discard_action:
         clear();
         return true;
      case R.id.circel_action:
         toggleCircel();
         return true;
      case R.id.loadfile:
         loadFile();
         return true;
      case R.id.stop: //only visible when playing an MP3 file
         stop();
         return true;
      default:
         return super.onOptionsItemSelected(item);
   }
}

//start playing selected MP3
```

```
	public void play(View v){
		if(!m_chosen.equals("")){
			try {
					mp.setDataSource(m_chosen);
					mp.prepare();
					mp.start();
					now.setToNow();
					drawView.starttime = now.format("%Y%m%d %H%M%S");
					long time= System.currentTimeMillis();
					drawView.millitime = time;
					mp.setOnCompletionListener(new OnCompletionListener() {
						@Override
						public void onCompletion(MediaPlayer mp) {
							now.setToNow();
							drawView.endtime = now.format("%Y%m%d %H%M%S");
							long time= System.currentTimeMillis();
							drawView.millistop = time - drawView.millitime;
							ImageButton playButton = (ImageButton) findViewById(R.id.play);
						playButton.setVisibility(View.VISIBLE);
						menu.findItem(R.id.stop).setVisible(false);
						mp.reset();
						}
					});
					ImageButton playButton = (ImageButton) v;
				playButton.setVisibility(View.GONE);
				menu.findItem(R.id.stop).setVisible(true);
			} catch (Exception e) {
					Log.e("log", e.toString());
				}


		}else
			Toast.makeText(CoreaffectActivity.this, "Geen geluidsbestand gekozen", Toast.LENGTH_LONG).show();

	}

	//stop the MP3-file currently playing
	public void stop(){
		try{
			ImageButton playButton = (ImageButton) findViewById(R.id.play);
			playButton.setVisibility(View.VISIBLE);
			menu.findItem(R.id.stop).setVisible(false);
			mp.stop();
			mp.reset();
		}catch (Exception e){
			Log.e("log", e.toString());
		}
	}

	//set value to show or hide a path created by touching the screen
	public void toggleTrace(View view) {
		drawView.tracetoggle = !drawView.tracetoggle;
		Toast.makeText(CoreaffectActivity.this, "Trace is nu: "+drawView.tracetoggle, Toast.LENGTH_LONG).show();
	}

	//set value to show or hide circle around the touched area
	public void toggleCircel() {
		drawView.circeltoggle = !drawView.circeltoggle;
		Toast.makeText(CoreaffectActivity.this, "Circel pointer is nu: "+drawView.circeltoggle,
Toast.LENGTH_LONG).show();
	}

	//create an input field to set the ID of the participant (added to file information)
	public void setName(final View view) {
```

```
        AlertDialog.Builder alert = new AlertDialog.Builder(this);
                alert.setTitle("Vul gegevens in");
                final EditText input = new EditText(this);
                input.setHint("vul proefpersoon ID in");
                alert.setView(input);
                alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                        String name = "" +input.getText();
                        drawView.name=name;
                        if(view.getId() == R.id.save)
                                save(view);
                }
                });
                alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                        Toast.makeText(CoreaffectActivity.this, "ID niet aangepast, bestand naam blijft gelijk",
Toast.LENGTH_LONG).show();
                }
                });
                alert.show();
  }

  //create an input field to set the name of the researchers (added to file information)
  public void setOnderzoekers(final View view) {
                AlertDialog.Builder alert = new AlertDialog.Builder(this);
                alert.setTitle("Vul gegevens in");
                final EditText input = new EditText(this);
                input.setHint("vul naam onderzoekers in");
                alert.setView(input);
                alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                        String name = "" +input.getText();
                        drawView.onderzoekers=name;
                        if(view.getId() == R.id.save)
                                save(view);
                }
                });
                alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                        Toast.makeText(CoreaffectActivity.this, "Naam onderzoekers niet aangepast, bestand naam blijft
gelijk", Toast.LENGTH_LONG).show();
                }
                });
                alert.show();
  }

  //call the save routine in DravView to save all the current data, and clear that data so we can start over
  public void save(View view) {
        menu.findItem(R.id.save_action).setEnabled(false);
        if(drawView.onderzoekers.equals("")){
                setOnderzoekers(view);
                return;}
        if(drawView.name.equals("")){
                setName(view);
                return;}
        String[] mp3a = m_chosen.split("/");
        String mp3 = mp3a[mp3a.length -1];

        DrawingView.writeToSDFile(drawView.onderzoekers,drawView.name,mp3,drawView.trace,drawView.starttime,dra
wView.endtime, drawView.millistop);
        drawView.clearAll();
        Toast.makeText(CoreaffectActivity.this, "resultaat opgeslagen", Toast.LENGTH_LONG).show();
        reEnable();
   }
```

```
    //the save button was made unclickable to prevent accidental doubleclick, reenable it
    public void reEnable(){
                        timeout.postDelayed(enableSave, 2000);
    }
    private Runnable enableSave = new Runnable(){
      public void run(){
                menu.findItem(R.id.save_action).setEnabled(true);
      }
    };

    //clear the screen
    public void clear(){
            drawView.clearAll();
            Toast.makeText(CoreaffectActivity.this, "huidige trace gewist", Toast.LENGTH_LONG).show();
    }

    //create a simple filedialog to open the MP3
            public void loadFile() {
                        SimpleFileDialog FileOpenDialog =  new SimpleFileDialog(CoreaffectActivity.this, "FileOpen",
                        new SimpleFileDialog.SimpleFileDialogListener(){
                                    @Override
                                    public void onChosenDir(String chosenDir){
                                                m_chosen = chosenDir;
                                    }
                        });
                        FileOpenDialog.Default_File_Name = "";
                        FileOpenDialog.chooseFile_or_Dir();
            }

}
---------------------------------------------------- DrawingView.java ----------------------------------------------
package nl.foaly.coreaffect;

import java.io.File;
import java.io.FileOutputStream;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Stack;
import android.content.Context;
import android.os.Handler;
import android.text.format.Time;
import android.util.AttributeSet;
import android.util.Log;
import android.view.View;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Path.Direction;
import android.graphics.PorterDuff;
import android.graphics.PorterDuff.Mode;
import android.graphics.PorterDuffXfermode;
import android.view.MotionEvent;


public class DrawingView extends View{

            //drawing path
            private Path drawPath, delPath;
            //drawing and canvas paint
            private Paint drawPaint, canvasPaint, delPaint;
            //initial color
```

```
private int paintColor = 0xFF660000;
//canvas
private Canvas drawCanvas;
//canvas bitmap
private Bitmap canvasBitmap;
//trace array
public ArrayList<String> trace = new ArrayList<String>();
//deletion Stacks
public Stack<coord> path = new Stack<coord>();
public Stack<coord> circel = new Stack<coord>();
//trace en circel Toggle
public boolean tracetoggle = true;
public boolean circeltoggle = false;
//name
public String name = "";
//onderzoekers
public String onderzoekers = "";
//clearer
public Handler clearer = new Handler();
//music timing
public String starttime = "", endtime = "";
public long millitime =0, millistop =0;
//sizes
int ws,hs;

//reset all values and clear screen
public void clearAll(){
		millitime =0;
		millistop =0;
		starttime = "";
		endtime = "";
		trace.clear();
		startNew();
}

//create Drawing Area
public DrawingView(Context context, AttributeSet attrs){
		super(context, attrs);
		setupDrawing();
}

//clear the screen from any trace
public void startNew(){
   drawCanvas.drawColor(0, PorterDuff.Mode.CLEAR);
   invalidate();
}

//declare paint and path values
private void setupDrawing(){
		//get drawing area setup for interaction
		drawPath = new Path();
		delPath = new Path();
		drawPaint = new Paint();
		drawPaint.setAntiAlias(true);
		drawPaint.setStrokeWidth(5);
		drawPaint.setStyle(Paint.Style.STROKE);
		drawPaint.setStrokeJoin(Paint.Join.ROUND);
		drawPaint.setStrokeCap(Paint.Cap.ROUND);
		drawPaint.setColor(paintColor);

		delPaint = new Paint();
		delPaint.setColor(Color.WHITE);
		 delPaint.setXfermode(new PorterDuffXfermode(Mode.CLEAR));
		 delPaint.setAlpha(0x255);
```

```
            delPaint.setAntiAlias(true);
            delPaint.setDither(true);
            delPaint.setStyle(Paint.Style.STROKE);
            delPaint.setStrokeJoin(Paint.Join.ROUND);
            delPaint.setStrokeCap(Paint.Cap.ROUND);
            delPaint.setStrokeWidth(6);
            canvasPaint = new Paint(Paint.DITHER_FLAG);
}

//when size changed, store the dimensions and make only the circle drawable
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);
        canvasBitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
        drawCanvas = new Canvas(canvasBitmap);
        Path clippath = new Path();
        ws = w;
        hs = h;
        clippath.addCircle(w/2,h/2,(float) (h/2.270720322),Direction.CW);
        drawCanvas.clipPath(clippath);

}

//add paint to canvas
@Override
protected void onDraw(Canvas canvas) {
        canvas.drawBitmap(canvasBitmap, 0, 0, canvasPaint);
        canvas.drawPath(drawPath, drawPaint);
        canvas.drawPath(delPath, delPaint);
}

//when the canvas is touched save the location and time of that touch and make add paint if applicable
@Override
public boolean onTouchEvent(MotionEvent event) {
        float touchX = event.getX();
        float touchY = event.getY();
        float adjustedX;
   float adjustedY;
        Time now = new Time();
        long time;
        long dif;

        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
                now.setToNow();
          if(tracetoggle){
                path.add(0,new coord(touchX, touchY));
                startPath(500);
          }
          time= System.currentTimeMillis();
          dif = time-millitime;
          adjustedX = (float) ((touchX - (ws/2))/(hs/22.689));
          adjustedY = (float) ((touchY - (hs/2))/(hs/22.689));
          trace.add(""+now.format("%H%M%S")+" [" + dif + "] -> "+adjustedX+ " - "+adjustedY);
          if(circeltoggle){
                drawCanvas.drawCircle(touchX, touchY, 70, drawPaint);
                circel.add(0,new coord(touchX,touchY));
          }
          drawPath.moveTo(touchX, touchY);
          break;
        case MotionEvent.ACTION_MOVE:
                now.setToNow();
                if(tracetoggle){
                        drawPath.lineTo(touchX, touchY);
```

```
                                path.add(0,new coord(touchX, touchY));
                                delPath(500);
                        }
                    if(circeltoggle && !circel.isEmpty()){
                        coord c = circel.pop();
                        drawCanvas.drawCircle(c.x, c.y, 70, delPaint);
                    }
                    if(circeltoggle){
                        drawCanvas.drawCircle(touchX, touchY, 70, drawPaint);
                        circel.add(0,new coord(touchX,touchY));
                }

                    drawCanvas.drawPath(drawPath, drawPaint);
                drawPath.reset();
                drawPath.moveTo(touchX, touchY);
                time= System.currentTimeMillis();
                    dif = time-millitime;
                    adjustedX = (float) ((touchX - (ws/2))/(hs/22.70720322));
                    adjustedY = (float) ((touchY - (hs/2))/(hs/22.70720322))*-1;
                    trace.add(""+now.format("%H%M%S")+" [" + dif + "] -> "+adjustedX+ " - "+adjustedY);
        Log.d("log",now.format("%H%M%S")+" [" + dif + "] -> "+adjustedX+ " - "+adjustedY);

                    break;
                case MotionEvent.ACTION_UP:
                        if(tracetoggle){
                                drawCanvas.drawPath(drawPath, drawPaint);
                        drawPath.reset();
                        resetPath(500);
                        }
                        if(circeltoggle && !circel.isEmpty()){
                            coord c = circel.pop();
                            drawCanvas.drawCircle(c.x, c.y, 70, delPaint);
                        }
                    break;
                default:
                    return false;
                }
                invalidate();
                return true;
        }


//remove the paint after i miliseconds
public void delPath(int i){
         clearer.postDelayed(delpath, i);
}

//remove the paint
private Runnable delpath = new Runnable(){
   public void run(){
        coord c = path.pop();
        delPath.lineTo(c.x, c.y);
        drawCanvas.drawPath(delPath, delPaint);
        //drawCanvas.drawCircle(c.x, c.y, 70, delPaint);
                delPath.reset();
                invalidate();
                delPath.moveTo(c.x, c.y);
                Log.w("del", "try del: "+c.x+" - "+c.y);
        invalidate();
   }
 };

        //remove the paint on first touch after i miliseconds
        public void startPath(int i){
```

```
                clearer.postDelayed(startpath, i);
        }

        //due how to Android deals with touch events remove the paint on first touch instead of move
        private Runnable startpath = new Runnable(){
        public void run(){
                coord c = path.pop();
                delPath.moveTo(c.x, c.y);
                invalidate();
        }
        };

        ///remove the paint when finger stops touching after i miliseconds/
   public void resetPath(int i){
                clearer.postDelayed(resetpath, i);
   }

   //due how to Android deals with touch events remove the paint when finger stops touching instead of move
   private Runnable resetpath = new Runnable() {
      public void run(){
        drawCanvas.drawPath(delPath, delPaint);
        delPath.reset();
        invalidate();
      }
   };

        //write the data collected by the touch events to file
        public static void writeToSDFile(String onderzoekers, String name, String mp3, ArrayList<String> arrayList, String
start, String stop, Long length){

          File root = android.os.Environment.getExternalStorageDirectory();
          Time now = new Time();
          now.setToNow();

          File dir = new File (root.getAbsolutePath() + "/core_affect_logs");
          dir.mkdirs();

          File file = new File(dir, now.format("%Y%m%d %H%M%S")+"#" + onderzoekers +" - "+name+"_"+mp3+".txt");

          try {
            FileOutputStream f = new FileOutputStream(file);
            PrintWriter pw = new PrintWriter(f);
            pw.println("begin trace"+ onderzoekers +" - "+name+"_"+mp3);
            pw.println("onderzoekers:" + onderzoekers);
            pw.println("proefpersoon:" + name);
            pw.println("mp3file: " + mp3);
            pw.println("start tijd:" + start);
            pw.println("stop tijd:" + stop);
            pw.println("length: " + length);
            pw.println("============================================================");
            for (String s : arrayList)
                        pw.println(s);
            pw.flush();
            pw.close();
            f.close();
            Log.i("log", arrayList.size()+" items weggeschreven voor test: "+mp3);

          } catch (Exception e) {
                Log.e("log", e.toString());
          }
        }
}
```

Appendix 4

In this appendix, you can find the timeline displaying the process of our bachelor project from the beginning of April to the middle of August:
Orange story points we did together, Miss Burger did the blue ones and Mister Veraart the green ones. Mister Veraart focused on the programming (appendix 1), while Miss Burger focused on the participants and performing the experiments (paper).